Everything from Scratch: Practical Ideas in an Organization without a Concept of Process Improvement

Hiroyuki Ito, SET taskforce, LINE, Japan

In this experience report, I present practical ideas to establish a new role, to improve product development teams, and to proceed company-wide problem-solving from scratch in the very strongly technology-oriented and fast-growing company without a concept of process improvement and clear supporters.

LINE Corporation [1] has been growing very rapidly and globally. However, the company had been struggling with increase of outages and they had given negative impacts to users and company's profits simultaneously. Since I joined LINE Corporation as the first member of “SET” (Software Engineer in Test) [2] in 2017, I and our team have been solving a wide variety of problems including reduction of outages, innovation in testing tools, onboarding, and so on by combining technical solutions and Agile methodologies based on my experiences as an Agile Coach. We have been pursuing "best for our users and business" fiercely and relentlessly with a series of experiments. Now, some teams in our company utilize our ideas that we reflected, experimented, and succeeded from scratch.

# Introduction

"LINE" is a free message, voice calls and video calls service for smartphones that has released since 2011. Our company name is derived from this service [3].

After the first release of “LINE”, LINE Corporation has increased its users and messages transferred rapidly and globally. Especially, its high-quality sound, and the "sticker" feature that we can send a variety of rich emoticons as a message attracted a lot of users.

For adapting to the rapid growth of LINE, we have been improving LINE's architectures and code base iteratively. We chose Microservice Architecture [4] to earn scaling out, independent development, and fast delivery capabilities.

However, increase of Microservices has been causing increase of outages simultaneously. The more we expand business areas like fintech, the more troubles at Integration Points [5] among each Microservice increase. Negative impacts to LINE users and company's profits became measurable in 2017.

LINE Corporation had struggling with solving these critical problems for themselves but failed. Because LINE Corporation was a strongly technology-oriented company and it didn't have a concept and experience of process improvement at that time. There were very few leaders and supporters to improve this situation.

Therefore, LINE Corporation had started looking for proper leaders externally. Evaluated over five years of my experiences and achievements as an Agile Coach and Test Automation engineer, I joined LINE Corporation as the first member of SET in September 2017.

# ESTABLISH SET ROLE BY ATTRACTING DECISION-MAKERS AND COLLEAGUES

## CHALLENGES

After joining LINE Corporation, I faced with tons of challenges to start my work as SET.

The biggest challenge was that a sense of crisis about increase of outages was not shared among employees. Only a few decision-makers like senior managers and executives were acknowledged and concerned about emergencies to solve negative impacts of outages to LINE users and company's profits as rapidly as possible.

Additionally, there were no clear ideas and solutions for increase of outages in the company. LINE Corporation had not experienced process improvements until then because of its very strongly technology-oriented fashion. Wide adoption to Microservice Architecture became a barrier to consider solutions beyond each service or technical silos.

Moreover, there were no shared understanding and consensus about SET role among decision-makers. LINE Corporation established SET job without clear objective, missions, and responsibilities. The company's intention at that time was very naive that just introducing Test Automation to the company might solve something.

## ACTIONS

My focus was to obtain consent from decision-makers and colleagues by discovering their real concerns and providing solutions iteratively and incrementally.

### SHARE A SENSE OF CRISIS BY VERBALIZING REAL CONCERNS

I thought not only decision-makers and colleagues but also I should know real problems LINE Corporation was facing with in a knowledgeable way. It was time to follow an ancient saying by Sun Tzu: "If you know the enemy and you know yourself, you will have almost 100 battles". From my experiences of Agile, I thought "Product Discovery" [6] might fit well.

Therefore, at first, I talked developers, QA persons, Product Managers, senior managers, and executives to hear their concerns and troubles directly. Through these conversations, I acknowledged that they had lots of non-verbalized problems. Additionally, I got that there were no persons who could verbalize these problems and share with other colleagues.

Next, I focused on helping colleagues to verbalize their concerns and to share them with other colleagues.

I investigated our services and products. I utilized "SonarQube" [7], a static code analysis tool, to know the code coverage and technical debts for each service. I also implemented and run some of unit and integration test scripts to know real behaviors of the products. Test scripts are good for understanding software under test [8] and finding problematic services.

Additionally, I focused on "outage reports". "Outage report" is a term which means both a published report and a postmortem meeting in our company. From these reports, I acknowledged that 1) reports were too technical to know impact on sales and profits, 2) reports didn't consider clear goals and actions to prevent the outages, and 3) "Channel Gateway", an aggregation service of our APIs towards external users, was the most problematic products.

After investigating services and outage reports, I could verbalize colleagues' concerns as follows: 1) increase of Channel Gateway's outages was the most critical issue that was giving negative impacts to external users, 2) failure detection of Channel Gateway took an average of 1 week and it was not acceptable for a Product Manager, and 3) testing APIs was insufficient in almost all of services because few persons knew how to test APIs programmatically. I shared these verbalized ones and agreed them with colleagues, managers and executives.

### GIVING IMPACTS CONTINUOUSLY TO INCREASE SUPPORTERS

For getting support from colleagues and decision-makers to proceed with radically new actions, I tried to "give impacts" [9] constantly to them.

From the first week I joined LINE Corporation, I achieved something and shared them with coworkers and decision-makers every week. Especially, I focused on providing working software, executable ones, and quantitative information.

Here is a list of achievements for my first 10 weeks.

|  |  |
| --- | --- |
| Week | Achievements |
| Week 1 | Started writing test scripts to understand services/products |
| Week 2 | Proposed first idea of SET activities to decision-makers |
| Week 3 | Built mechanism to run and report static code analysis regularly |
| Week 4 | Shared with developers how to build static code analysis |
| Week 5 | Proposed milestones of activities to decision-makers |
| Week 6 | Agreed with proposals/milestones with decision-makers |
| Week 7 | Collected information and tools of QA/Tests in one place |
| Week 8 | Implemented failure detection mechanism for Channel Gateway |
| Week 9 | Guided to start regular meetings with developers and QA persons |
| Week 10 | Started solving problems by developers step by step |

Figure 1. The list of achievements for my first 10 weeks as SET.

When I gave impacts, I utilized "3 KPIs": Sales, Profit, and Employee Satisfaction. It is that my former supervisor told me as a way to measure every business. For example, my first proposal of SET activities to decision-makers included reducing MTTR (Mean Time to Repair) of Channel Gateway by implementing proper failure detection mechanism for reducing negative impacts to external users. Additionally, I not only reported results of static code analysis, but also shared with developers how to build static code analysis mechanism from Employee Satisfaction aspect.

As a result, many developers started using my impacts like static code analysis mechanism and giving me concrete advices about problematic points of architectural design, operational difficulties, and so on. Their advices became good information sources to communicate with decision-makers. Additionally, decision-makers started taking time to define SET role with me. They couldn't ignore my impacts and developers' supports to SET.

### ITERATIVE AND INCREMENTAL CONSENSUS

In parallel with giving impacts, I built consensus on SET role with decision-makers gradually. From the first week I joined LINE Corporation, I started proposing ideas of SET; objective, missions, responsibilities, solutions, and milestones; iteratively based on gathered information I verbalized and got from colleagues.

This phase was analogous to start-up business or building new services. Through discussions, we corrected ideas of SET role, and built feeling of trust each other step by step. It was a weekly cycle of build-propose-learn. I named it as "Iterative and Incremental Consensus".

As a result, within 45 days since I joined LINE Corporation, we agreed on 1) making failure detection faster and reducing MTTR by utilizing Test Automation techniques, 2) increasing API test scripts, and 3) focusing on Channel Gateway at first as SET role, goals, and milestone. It was a result from scratch.

## RETROSPECTIVE

Establishing SET role in LINE Corporation was analogous to start-up business or building new services to me.

In an organization without a concept of process improvement, showing concrete examples to improve small things step by step could give impacts to the organization positively. Verbalizing colleagues' concerns as Product Discovery became good information sources. Additionally, providing achievements and building feeling of trust as impacts worked for getting support from colleagues and decision-makers. Moreover, Iterative and Incremental Consensus was effective for collaboration and quick agreement on totally new ideas with decision-makers. Through these activities, 3 KPIs worked as common criteria for improvements among colleagues, decision-makers, and I.

# INNOVATE SOLUTIONS BY EXPERIENCING HARDSHIPS TOGETHER

## CHALLENGES

After establishing SET role, I started actions as SET by obtaining consent from colleagues and decision-makers. After 6 months, we hired new SETs and formed a team of SET. I thought we could proceed our activities more quickly and widely, however, we faced with some new obstacles.

One obstacle was that the failure detection mechanism we implemented for Channel Gateway didn't become established in the team. At first, we built the failure detection system by combining API test scripts, running them via CI servers periodically, and notifying errors and failures to the team members quickly. We used JUnit, Spring Boot, Jenkins, and Slack to meet the team's skill sets. After providing the system including manuals to the team, it worked well for the first 2 months. The team could detect failures within 1 hour. Some developers started implementing test scripts. However, team members started ignoring notifications from the system soon without clear reasons.

Another obstacle was that performance problems at Sticker Shop had emerged. They used one open-sourced performance testing tool. However, it couldn't provide enough capabilities to detect emerging issues. The team tried to improve the situation but failed. It became urgent issues in our company at that time.

Both obstacles had a commonality that we couldn't approach each team's real problems properly due to lack of knowledge of their contexts deeply. Just providing tools and guidelines as consultants won't work in both cases. We often got requests from lots of teams to provide standardized tools, guidelines, and reference implementations. However, we never saw that they worked fine and solved their core problems because they tend to be far from the team's real needs and contexts.

## ACTIONS

Our choice was to join each team and work together for understanding each team's contexts, finding proper solutions, and committing the teams and solutions more.

### REFINED THE FAILURE DETECTION SYSTEM WITH KARATE

For the failure detection system of Channel Gateway's team (hereinafter called "the team"), we utilized Product Discovery again. We started direct conversations with the team members to hear and verbalize their real concerns and needs at first. The team and we SET team talked every day as deeply as possible by using video conference system because each team has been working at different offices. Additionally, we discussed with the team's Product Manager if he came to our office.

Through these discussions, we understood that test scripts written in JUnit were hard to read, write, and maintain for the team members. Therefore, the team and we SET team looked for and evaluated proper tools which would meet our needs together.

After a series of evaluations, we chose "Karate" [10], an open-sourced framework which focuses on API testing with BDD (Behavior-Driven Development) style and Gherkin format. The team members favored its readability, maintainability, and extensibility.

After the decision to use Karate, the team and we SET team started rewriting all test scripts from JUnit to Karate collaboratively. At first, we wrote examples of test scripts with Karate for teaching the team members how to implement tests. Next, the team and we SET team had been tackling with wide variety of testing issues together; product's features that were hard to test, preparation of test data, prioritization of testing, extension of Karate features, and so on.

After 3 months' collaborative work, finally the failure detection system with Karate became established in the team. Now all of the team members including the Product Manager are writing and maintaining test scripts with Karate routinely without our support.

### CREATED NEW TOOLS THAT FIT TEAM'S CONTEXT

For Shopping features, we did the same approach as public APIs to discover their real needs and concerns at first.

We found that improving to use the existing in-house performance testing tool was impractical. It could produce only 10% of loads what we wanted to test. It was not easy to expand and/or modify features. Additionally, most of the product development team's members were familiar with Kotlin language. Implementing test scripts with Groovy was hard for them. Moreover, usage of Docker [11] and Kubernetes [12] were expanding at that time in our company. We thought it was a good chance to utilize these new tools and approaches to improve our performance testing.

Therefore, we decided to create a new in-house performance testing tool named "Ayaperf". Ayaperf is a Java wrapper of Locust [13] that can use Kubernetes to increase loads easily with enough volume. Developers can write test scripts of performance testing with Java and Kotlin. We did iterative and incremental style to implement and improve Ayaperf with Shopping features’ developers. After 3 months' collaborative work, finally Ayaperf became stable. Developers started detecting performance issues with it before release. Additionally, they could correct issues by themselves without hurting production code. They found and solved 3 hidden performance issues by utilizing Ayaperf in one month.

### PRACTICED PROCESS IMPROVEMENTS WITH TEAMS

At public APIs and Shopping features, we found the effectiveness of working with product development teams to find their real needs and solve them. This approach worked well. However, I thought it was not enough and sufficient. I saw that lots of teams stopped solving problems by themselves after coaches left teams. It is a failure if improvements don't continue after coaches' left. Therefore, I expanded our activities to making product development process improvement as a habit especially at public APIs.

We had found and solved issues as homework every week. We had continued applying new Karate features and refactoring test scripts. Additionally, we had implemented a notification mechanism via Slack [14] to reduce MTTR. Moreover, we had asked product development team members for clarifying objectives, quantitative values they will provide to users, and rough milestones of each task every week. We utilized the idea of Scrum framework to make continuous improvement as a habit of the team. We had continued these activities for about 3 months.

After that, the product development team became able to clarify quarterly milestones, prioritize tasks based on business values, and improve test scripts and the failure detection system by their own. They reduced about 30% of outages. They really became the self-organized team. Finally, we stopped supporting the team.

## RETROSPECTIVE

We could solve essential problems and improve processes of each product development team by working collaboratively and deeply with them. We SETs and product development teams implemented Test Automation and related techniques based on the idea of Product Discovery. Additionally, each team becomes sophisticated. For example, the Product Manager of public APIs writes test scripts with Karate routinely. He often says that the Product Manager may disturb the team by writing production codes, but can contribute to the team by writing test scripts! He is utilizing test scripts to understand behavior of the product deeply, to clarify next actions and goals of the product and the team, and to guide team members doing Developer Testing for Build Quality In.

Additionally, we learned a lot of things to improve our approaches through working with them. The consulting-style approach is useful to keep the whole image of activities, however, we cannot approach essential problems. On the other hand, the working-together approach is effective to discover and solve essential problems quickly, but we may lose the whole image of activities because of too focusing on one product development team. Therefore, we should utilize both styles based on the phase of activities.

The English word "compassion" derives from Latin's "compati", which means "suffer with". We think this is the point of leading and guiding new things like Agile. We SETs and product development teams had been suffering from the same problems by working together. We had considered solutions and solved problems one by one together. These series of activities had constructed real collaborative relationships. Moreover, these relationships had become boosters for adapting to new Test Automation tools and process improvements.

# PROCEED COMPANY-WIDE PROBLEM-SOLVING AS TRANSFORMATIONAL LEADERS

## CHALLENGES

We had been solving a lot of technical and process issues of each product by working together with each product development team's members. These activities and achievements have been recognized as huge successes by executives. However, these successes had led us SET team to the next level of challenges.

The first challenge was the company-wide strategic and management problems. A lot of product development teams could not show their own missions, goals, plans, and milestones to decision makers like senior managers and executives beforehand. Additionally, these teams couldn't share their current status and problems in a timely manner. Decision makers had been frustrating that they couldn't make decisions properly and precisely. On the other hand, we SET team had been showing that information timely from the beginning of all activities. Therefore, decision makers requested us SET team to teach product development teams to express that information properly.

The second challenge was about onboarding. In July 2019, our team had 4 members and we hired one recent graduate and one mid-career employee simultaneously as new SETs. To proceed our activities smoothly, we needed to make onboarding as the top priority task.

The third one was a doubt about testing and quality assurance. LINE Corporation hires Test Engineers and QAs. However, most of them had only been doing End-to-end testing manually via client applications. In the era of Microservices, I thought it is not practical to detect bugs and solve them beforehand with these activities. Additionally, most of them didn't care about deployment, release, and contribution to our business. They were just interested in doing their own tasks by developer's requests. Such behavior was not what we SET aimed to do. On the other hand, we named our role as "Software Engineer in Test". The word "Test" made our colleagues misunderstand that we SET were the same as Test Engineers and QAs. The notion of testing and quality assurance were just a burden and a constraint that narrowed our activities to improve services and products. Therefore, I thought we needed different approaches to change these assumptions drastically.

## ACTIONS

For solving these totally different challenges, we started lots of actions including not only Test Automation and technical ones, but also engineering management, education, innovation, and so forth.

### LEAD ENGINEERING MANAGEMENT IMPROVEMENT

For solving the company-wide strategic and management problems, we had started showing our activities and installing our ways into other teams. In other words, we had started leading engineering management improvement based on decision makers' demands.

At first, we shared our milestones with other teams over and over again as an example of engineering management strategy and planning. Additionally, we held workshops for these teams to support their planning, defining mission, reporting, and so forth. For example, I held the Drucker Exercise [15] and the User Story Mapping [16] workshops to one team for teaching the idea of product ownership. After these activities, some teams started defining their own milestones and sharing them to decision makers in a timely manner.

On the other hand, we attended other teams' meetings to improve. If the meeting was full of verbose and meaningless reporting without any decision making and productive communication, we proposed rules like reporting only necessary for decision making and applying timeboxing. We often utilized the idea of Impact Meeting [17] by Mike Cohn. Moreover, we stopped some meetings that couldn't provide any value. Clarified mission and milestones were useful to distinguish whether the meeting was valuable or not. We could use clear mission and milestones as the pointer of conversation as the same as the User Story.

### LEARNING SESSION

For proceeding our onboarding smoothly, I decided to utilize the idea of "Learning Session". Learning Session is a way of study sessions during business hours taught by Chris Lucian at Agile2017 [18].

Here are basic rules. We have been holding Learning Session basically for 30 to 60 minutes every day during business hours as a work. We can choose any topics we assume it's necessary for our daily work. Anyone can speak and facilitate it with Mob Programming way with a fun and without criticism.

Through a series of Learning Sessions, we have been learning a wide variety of tools, techniques, process improvements, and so forth. We learned Karate framework. All team members can set up it, write test scripts, run tests, and teach them to other persons. We became accustomed to shortcut keys of IntelliJ IDEA, JIRA, and Confluence. We often review programs via GitHub's Pull Requests together. We refactor test scripts with learning test and architectural design techniques. We frequently demonstrate our work-in-progress tasks to get feedbacks quickly. Moreover, we experiment process improvements like Scrum, Kanban, the Drucker Exercise, and so on.

As a result, we smoothly finished onboarding for two newcomers. They could write programs, pass reviews, and deploy their programs within 3 days. They could adapt to our team's rules and culture, like demonstrating their results to users very frequently for getting fast feedbacks, focusing on release, and experimental work style very quickly. They became contributing to our products and services within 1 or 2 months.

Moreover, we found that Learning Session gave 3 additional impacts to our team.

The first impact was the growth of our team, not only of newcomers. We often shared each work among team members. We frequently solved each member's problem together. As a result, all team members could substitute other's works. We can say we have been doing “handover” every day. We could increase the "Truck Number" coined by Jim Coplien and enhance our team's capabilities to solve problems.

The second impact was the psychological one. We were accustomed to show work-in-progress tasks and get feedbacks. It made us easier to ask questions and discuss solutions. We could propose, accept, and try new ideas without fear. Additionally, we often drilled trouble shootings and recovering services as Learning Session to acquire skills of Psychological Safety. We have been reducing psychological pressures by atmosphere, mechanisms, and skills.

The third and last impact was for evaluation. We could reduce the burden of personnel evaluation dramatically for both an evaluator and a member to be evaluated. On one hand, I, an evaluator, can touch members' activities, achievements, and impediments directly everyday through Learning Sessions. Therefore, I can evaluate each member quickly, easily, and properly every day. On the other hand, members can appeal their achievements to me very easily. Additionally, we can adjust behavior each other through daily observations and feedbacks to meet the team's objective. We didn't need to set evaluation meetings at once around the evaluation period and it saved our time and resources. We can say it as an Agile way of evaluation and human resource management.

### TEST AUTOMATION FOR RESILIENCE

For overcoming the limitation of testing and quality assurance in the era of Microservices, we decided to shift our focus to resilience, deployment, and release rather than detecting and solving bugs beforehand.

At first, we started combining Karate framework with Zipkin [19], a distributed tracing system. Our failure detection system with Karate was good at fast detection of failures and outages. However, it could not pinpoint a root cause in a fleet of Microservices. This was an emerging problem for Product Managers at that time. Therefore, we aimed to make our failure detection system more intelligent.

Our approach was to show tracing information of each Microservice on our test report by adding Zipkin's trace ids to call APIs to test. This test report could pinpoint which Microservice failed by utilizing Zipkin's trace ids. It means that we can pinpoint a Product Manager who is responsible for failed Microservice. Additionally, it can reduce MTTR more and save other Product Managers' time. Our approach was utilizing the idea of observability and monitoring via Test Automation. We named this report as "Sebas Report". (The name "Sebas" is derived from a famous butler like Jenkins.)

After the release of Sebas Report, we started promoting Karate and Sebas Report company-wide. Additionally, we started recommending to each product development team to utilize not only reducing MTTR, but reducing lead time for changes, and increasing deployment frequency as KPIs to measure improvement and productivity. I utilized the idea of Four Key Metrics [20] as a way of contribution to our business. After these activities, some teams stopped blindly relying on QAs and enhanced the ratio of Developer Testing.

## RETROSPECTIVE

We expanded our activities toward engineering management improvement based on decision makers' demands. Additionally, we experimented new ideas like Learning Session and utilizing Test Automation for resiliency. Through these activities, we have been redefining our goals and responsibilities based on continuous experiments to contribute to our business. We can say we transformed us as a team of Transformational Leaders.

# LESSONS LEARNED

Through these series of activities, we learned three new ideas.

At first, Agile methodologies worked for starting up new roles and teams. Product Discovery, Iterative and Incremental Consensus, and showing results iteratively attracted colleagues and decision makers. It helped SET team's starting up a lot.

Second, working closely with product development teams was very effective for improving processes and achieving missions. Just providing guidelines and references to product development teams didn't work. Showing working examples enriched communication. Technical excellence was a necessary piece to provide examples and solutions properly. We utilized technology as a communication driver, however, only technology was not enough. We should leverage communication with both technical excellence and Agile methodologies.

Third, it was necessary to change our responsibilities continuously. We have been changing and expanding our responsibilities from Test Automation, engineering management improvement, innovations, to Transformational Leaders. These actions are for contributing to our business based on continuous experiments.

We SET team have been finding and solving problems gradually and extensively by combining technical excellence and Agile methodologies, and adjusting our responsibilities for contributing to our business.

# WHAT'S NEXT?

Currently, we have been trying and investigating the following ideas for further improvements.

The first one is building "Testable and Disposable Infrastructure". Currently, we don't have enough production-like test environments. It makes us hard to test enough to detect bugs and reproduce outages quickly without fear for operation miss and misconfiguration. It becomes a huge barrier for testing. Therefore, we have been trying to build production-like test environments with Container and its orchestration mechanism like Docker and Kubernetes for making testing easier without fear. We are aiming to provide immutable and disposable containers, and a way of building their relationships easily. We named this idea and mechanism as Testable and Disposable Infrastructure. As a preparation, we are adapting to Testcontainers [21] for testing persistence layer like RDBMS and NoSQL.

The second one is to organize ideas, terminology, and techniques of Test Automation. There are lots of confusion about the difference among Unit Testing, Integration Testing, API Testing, End-to-end Testing, and so on. It is more than Microservices. Therefore, we started summing up these points as a guideline and reference implementations. We clarified how to distinguish Test Levels and design each test as a guideline. Additionally, we implemented and shared reference implementations of them company-wide. Moreover, we held Hackathon events for teaching Karate. At the first event, attendees could implement tests for their product's APIs within 2 hours.

The last one is experimenting Design Sprint [22] for solving complex technical problems at a brownfield product. We have been working with one product development team which develops and operates mature product. There are lots of problems, ideas to solve them, and huge confusion. We started utilizing an idea named Design Sprint to clarify each problem, prioritize each idea, and experiment whether the idea is valuable or not. We try new ideas within 1-week cycle. We gather feedbacks and decide whether to proceed the idea or switch to other one within this short cycle. We continue this inspection and adaptation approach until solving problems. Currently, this team is adapting to Testcontainers and Testable and Disposable Infrastructure with this approach.

# CONCLUSIONS

There have been lots of problems. Microservice Architecture is useful for independent develop-ability and deployability, however, it tends to strengthen apathy toward other teams and services/products. Additionally, there are short of leaders who can find and solve problems beyond silos and teams. Moreover, quality assurance approach is not proper for solving outages of Microservices.

We SET team have been solving these problems as follows.

At first, we aim to overcome limitations of feature teams for Microservice Architecture. Many Agile enthusiasts and practitioners insist of the importance of a feature team rather than a component team. However, only applying for a feature team cannot tame Microservice Architecture. Therefore, SET team behaves to tie each product development team, service, and product with providing a whole image of business, and ways to test and recover each service.

Second, we train and nurture leaders who can find problems and solve them beyond teams and silos. LINE Corporation has lots of excellent software engineers. However, there are few leaders. Only technical excellence cannot tame complexity of Microservice Architecture and our business. SET team is effective for nurturing these leaders by utilizing both technical skills and Agile methodologies.

Third, we have been building and expanding tools, process improvements, and new KPIs to get over limitations of quality assurance. The combination of Karate framework and Sebas Report makes testing and recovering Microservices easier. Additionally, the idea of Build Quality In is becoming common for some services/products. Focusing on profitability and MTTR rather than the number of bug detection before releases works now in organizational performance and culture perspective.

SET is derived from Google. This role is responsible for enhancing productivity of engineers by utilizing test automation, automation infrastructure, and process improvements in Google. SET in LINE Corporation tried to start from this definition. However, we have been changing responsibilities continuously for pursuing contribution to our company's business performance. We have been expanding our responsibility from Test Automation to company-wide process improvements with continuous impacts to stakeholders.

Currently, we are transforming ourselves as a team of Transformational Leaders. SET team defined its philosophy as "WOW DX": the combination of "WOW", LINE Corporation’s action philosophy and mindset [23], and the Developer eXperience. We are pursuing a great Developer eXperience for all of product development team members and stakeholders in LINE Corporation with automation techniques and Agile methodologies. We continue to pursue improving all of our business relentlessly.

REFERENCES

[1] LINE. https://linecorp.com/en/.

[2] Whittaker, J. Arbon, J., & Carollo, J. 2012. How Google Tests Software. Addison-Wesley Professional.

[3] LINE. https://linecorp.com/press/2013/0401472.

[4] Fowler, M. 2014. Microservices. https://martinfowler.com/articles/microservices.html.

[5] Nygard, M. 2018. Release It!: Design and Deploy Production-Ready Software 2nd Edition. Pragmatic Bookshelf.

[6] Hussman, D. 2015. Product Discovery On A Single Page. http://productdiscoverycanvas.com/tag/david-hussman/.

[7] SonarQube. https://www.sonarqube.org/.

[8] Whittaker, J. Arbon, J., & Carollo, J. 2012. How Google Tests Software. Addison-Wesley Professional.

[9] Whittaker, J. Arbon, J., & Carollo, J. 2012. How Google Tests Software. Addison-Wesley Professional.

[10] Karate. https://github.com/intuit/karate.

[11] Docker. https://www.docker.com/.

[12] Kubernetes. https://kubernetes.io/.

[13] Locust. https://locust.io/.

[14] Slack. https://slack.com/.

[15] The Agile Warrior. https://agilewarrior.wordpress.com/2009/11/27/the-drucker-exercise/.

[16] Jeff Patton & Associates. https://www.jpattonassociates.com/user-story-mapping/.

[17] Mountain Goat Software. https://www.mountaingoatsoftware.com/blog/four-questions-to-fix-low-attendance-at-your-sprint-reviews.

[18] Lucian, C. 2017. Growing the Mob. https://www.agilealliance.org/wp-content/uploads/2017/02/GrowingTheMob.pdf.

[19] Zipkin. https://zipkin.io/.

[20] ThoughtWorks. https://www.thoughtworks.com/radar/techniques/four-key-metrics.

[21] Testcontainers. https://www.testcontainers.org/.

[22] GV. https://www.gv.com/sprint/.

[23] LINE. https://linecorp.com/en/company/mission.

[1] Transformational leadership. https://en.wikipedia.org/wiki/Transformational\_leadership.

[2] Forsgren, N. Humble, J., & Kim, G. 2018. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations. IT Revolution Press.